CONTROLLER

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package rcw5k2mvctimerstopwatchfxmls21;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import java.net.URL;

import java.text.DecimalFormat;

import java.util.Optional;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.chart.AreaChart;

import javafx.scene.chart.CategoryAxis;

import javafx.scene.chart.LineChart;

import javafx.scene.chart.NumberAxis;

import javafx.scene.chart.XYChart;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TextInputDialog;

import javafx.scene.image.ImageView;

import javafx.scene.text.Text;

/\*\*

\* FXML Controller class

\*

\* @author raymondwaidmann

\*/

//step 2: controller needs to implement the property change listener interface!!!!

public class Rcw5k2MVCTimerStopwatchFXMLS21ViewController implements Initializable, PropertyChangeListener {

@FXML

private ImageView dialImageView;

@FXML

private ImageView handImageView;

@FXML

private Text timeText;

@FXML

private Text timerText;

@FXML

private Text lapText;

@FXML

private Text avgLapText;

@FXML

private Button startStopButton;

@FXML

private Button recordResetButton;

@FXML

private LineChart<?, ?> recordChart;

@FXML

private AreaChart<?, ?> averageChart;

@FXML

private NumberAxis recordYAxis;

@FXML

private CategoryAxis recordXAxis;

@FXML

private NumberAxis averageYAxis;

@FXML

private CategoryAxis averageXAxis;

Rcw5k2MVCTimerStopwatchFXMLS21AnalogModel model;

//properly located in controller! Wergeles office hours;

private XYChart.Series recordSeries;

private XYChart.Series averageSeries;

/\*\*

\* Initializes the controller class.

\*/

@Override

public void initialize(URL url, ResourceBundle rb) {

model = new Rcw5k2MVCTimerStopwatchFXMLS21AnalogModel();

model.addPropertyChangeListener(this); //this is ok here and not at the bottom; Wergeles office hours

handImageView.setRotate(0);

recordSeries = new XYChart.Series();

averageSeries = new XYChart.Series();

initializeCharts();

recordChart.getData().add(recordSeries);

averageChart.getData().add(averageSeries);

textDialog();

model.setupTimer();

}

@FXML

private void startStopButton(ActionEvent event) {

if(model.isRunning()){ //STOP

model.stop();

recordResetButton.setText("Reset");

startStopButton.setText("Start");

startStopButton.setStyle("-fx-background-color: #00ff00");

} else { //START

model.start();

recordResetButton.setText("Record");

startStopButton.setText("Stop");

startStopButton.setStyle("-fx-background-color: #ff0000");

}

}

@FXML

private void recordResetButton(ActionEvent event) {

if(model.isRunning()){ //RECORD

if(!model.getTimesUp()){

model.record();

} else {

showError("Time is up... No more records...");

}

} else { //RESET

model.reset();

}

}

@Override

public void propertyChange(PropertyChangeEvent evt) {

if(evt.getPropertyName().equals("Analog")){

handImageView.setRotate(Double.parseDouble(evt.getNewValue().toString()));

} else if(evt.getPropertyName().equals("Digital")){

timeText.setText(evt.getNewValue().toString());

} else if(evt.getPropertyName().equals("Timer")){

timerText.setText(evt.getNewValue().toString());

} else if(evt.getPropertyName().equals("Lap")){

lapText.setText(evt.getNewValue().toString());

} else if(evt.getPropertyName().equals("AvgLap")){

avgLapText.setText(evt.getNewValue().toString());

} else if(evt.getPropertyName().equals("ResetButtonsAndImage")){

startStopButton.setText("Start");

startStopButton.setStyle("-fx-background-color: #00ff00");

recordResetButton.setText("Record");

handImageView.setRotate(0);

} else if(evt.getPropertyName().equals("RecordChartYAxis")){

recordYAxis.setUpperBound(Double.parseDouble(evt.getNewValue().toString()));

} else if(evt.getPropertyName().equals("AverageChartYAxis")){

averageYAxis.setUpperBound(Double.parseDouble(evt.getNewValue().toString()));

} else if(evt.getPropertyName().equals("RecordChart")){

recordSeries.getData().add(new XYChart.Data(evt.getOldValue().toString(), evt.getNewValue()));

} else if(evt.getPropertyName().equals("AverageChart")){

averageSeries.getData().add(new XYChart.Data(evt.getOldValue().toString(), evt.getNewValue()));

} else if(evt.getPropertyName().equals("ResetCharts")){

recordSeries.getData().clear();

recordChart.getData().clear(); //https://stackoverflow.com/questions/12093556/javafx-2-x-how-to-remove-xy-line-chart-once-plotted/13053403

recordChart.getData().add(recordSeries);

averageSeries.getData().clear();

averageChart.getData().clear(); //https://stackoverflow.com/questions/12093556/javafx-2-x-how-to-remove-xy-line-chart-once-plotted/13053403

averageChart.getData().add(averageSeries);

initializeCharts();

} else if(evt.getPropertyName().equals("TextDialog")){

textDialog();

}

}

//Properly located in Controller! Wergeles Office Hours

public void showError(String message){

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Alert");

alert.setHeaderText("Error!");

alert.setContentText(message);

alert.showAndWait();

}

public void initializeCharts(){

recordChart.setAnimated(false);

recordYAxis.setAutoRanging(false);

recordYAxis.setLowerBound(0);

recordYAxis.setUpperBound(model.getYRecord());

averageChart.setAnimated(false);

averageYAxis.setAutoRanging(false);

averageYAxis.setLowerBound(0);

averageYAxis.setUpperBound(model.getYAverage());

}

public void textDialog(){

//Getting User Input for the timer

TextInputDialog startTimeDialog = new TextInputDialog(); //https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextInputDialog.html

startTimeDialog.setTitle("Timer Start Time Set Up");

startTimeDialog.setHeaderText("Set up the start time:");

startTimeDialog.setContentText("Please set up the start time (Integer):");

boolean legalTime = false;

int tempTime = 0;

while(!legalTime){ //this loop went over with Xin during the developer night

Optional<String> input = startTimeDialog.showAndWait(); //showAndWait method returns the input entered bu the user or NULL if nothing is entered

if(input.isPresent()){

if((input.get().matches("[0-9]+") && input.get().length() > 0) == false){

showError("Insert a Positive Integer!");

} else{

tempTime = Integer.parseInt(input.get());

model.setTimerTime(tempTime);

legalTime = true; //need a setter for this as well

}

} else { //user cancels or closes out of input dialog

System.exit(0); //https://www.geeksforgeeks.org/system-exit-in-java/

}

}

if(tempTime > 60 || tempTime ==0) { //if input time is greater than 60, time is always up

model.setTimesUp(true);

timerText.setText("Times Up!");

} else {

DecimalFormat df = new DecimalFormat("00");

timerText.setText("Timer: " + df.format(tempTime) + ".00");

}

}

}

MODEL

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package rcw5k2mvctimerstopwatchfxmls21;

import java.text.DecimalFormat;

import javafx.animation.Animation;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.event.ActionEvent;

import javafx.util.Duration;

/\*\*

\*

\* @author raymondwaidmann

\*/

//step 1: extends abstractmodel !!!

public class Rcw5k2MVCTimerStopwatchFXMLS21AnalogModel extends Rcw5k2MVCTimerStopwatchFXMLS21AbstractModel {

DecimalFormat df = new DecimalFormat("00");

DecimalFormat doubleDf = new DecimalFormat("00.00");

private double secondsElapsed;

private final double tickTimeInSeconds; //how to change the resolution of the watch (make it smooth)

private final double angleDeltaPerSeconds; //degrees per second

private double rotation;

private int ticks;

private Timeline timeline;

private int minute;

private int second;

private int centisecond;

private int recordCounter;

private int recordTicks;

private int recordsSecond;

private int recordsCentisecond;

private double average;

private int timerTime;

private int timerSecond;

private int timerCentisecond;

private boolean timesUp;

private int yRecord;

private int yAverage;

public Rcw5k2MVCTimerStopwatchFXMLS21AnalogModel(){

secondsElapsed = 0.0;

tickTimeInSeconds = 0.01;

angleDeltaPerSeconds = 6;

ticks = 0;

recordCounter = 0;

average = 0.0;

timerTime = 0;

timesUp = false;

yRecord = 5;

yAverage = 5;

timerTime = 0;

timerSecond = 0;

timerCentisecond = 0;

}

public void setupTimer(){

//creating an animation (Keyframe and Timeline; 2 requirements to make)

KeyFrame keyFrame = new KeyFrame(Duration.millis(1000 \* tickTimeInSeconds),

(ActionEvent event) -> { //every 1000 ms a new keyframe is created

update();

});

timeline = new Timeline(keyFrame);

//set cycle count; call play method (2 requirements to run animation)

timeline.setCycleCount(Animation.INDEFINITE); //can use Animation.INDEFINITE for infinite cycles

}

public void update(){

ticks+=(100\*tickTimeInSeconds); //total number of centiseconds, used to calculate centiseconds, seconds, and minute

//rotating the hand on the stopwatch

secondsElapsed += tickTimeInSeconds;

double oldRotation = rotation;

rotation = secondsElapsed \* angleDeltaPerSeconds; //360 degrees, 60 tick marks, each tick 6 degrees

firePropertyChange("Analog", oldRotation, rotation);

//display values for digital output

centisecond = ticks%100;

second = (ticks/100)%60;

minute = ticks/6000;

String newDigital = (df.format(minute) + ":" + df.format(second) + "." + df.format(centisecond));

firePropertyChange("Digital", null, newDigital);

//display values for records

//Note that recordTicks gets reset to 0 whenever the record button is pressed

recordTicks+=(100\*tickTimeInSeconds);

recordsCentisecond = recordTicks%100;

recordsSecond = (recordTicks/100)%60;

//display values for timer

timerCentisecond = 100 - centisecond;

timerSecond = timerTime - second - 1;

if(timerCentisecond == 100) { //edge conditions require slight modifications

timerCentisecond = 0;

if (timerSecond != -1) timerSecond += 1; //if statement required to make sure that timer ends at the correct time

}

String newTimer = ("Timer: " + df.format(timerSecond) + "." + df.format(timerCentisecond));

if(!timesUp){

firePropertyChange("Timer", null, newTimer);

if (timerSecond == -1){ //timer reaches 0

if (timerCentisecond == 0){

firePropertyChange("Timer", null, "Times Up!");

timesUp = true;

}

}

}

}

public boolean isRunning(){

if(timeline != null){

if(timeline.getStatus() == Animation.Status.RUNNING){

return true;

}

}

return false;

}

public void start(){

timeline.play();

}

public void stop(){

timeline.pause();

}

public void record(){

//average calculation

average \*= recordCounter;

double doubleLap;

if (recordsCentisecond >= 10) { //without this if statement, lap time of 2.08 would show as 2.80 since recordCentisecond would simply be 8.

doubleLap = Double.parseDouble(recordsSecond + "." + recordsCentisecond);

} else {

doubleLap = Double.parseDouble(recordsSecond + ".0" + recordsCentisecond);

}

average += doubleLap;

recordCounter++;

average /= recordCounter;

String newLap = ("Lap: " + String.format("%02d", recordsSecond) + "." + String.format("%02d", recordsCentisecond));

firePropertyChange("Lap", null, newLap);

String newAvgLap = ("Avg. Lap: "+ doubleDf.format(average));

firePropertyChange("AvgLap", null, newAvgLap);

//while loops adjust the yaxes of the charts if necessary

while(doubleLap >= yRecord){

yRecord += 5;

firePropertyChange("RecordChartYAxis", null, yRecord);

}

while(average >= yAverage){

yAverage += 5;

firePropertyChange("AverageChartYAxis", null, yAverage);

}

firePropertyChange("RecordChart", Integer.toString(recordCounter), doubleLap);

firePropertyChange("AverageChart", Integer.toString(recordCounter), average);

recordTicks = 0;

}

public void reset(){

firePropertyChange("Digital", null, "00:00.00");

firePropertyChange("Lap", null, "Lap: 00.00");

firePropertyChange("AvgLap", null, "Avg. Lap: 00.00");

firePropertyChange("ResetButtonsAndImage", null, null);

average = 0;

yRecord = 5;

yAverage = 5;

if(ticks!=0){

firePropertyChange("Timer", null, "Timer: --:--");

firePropertyChange("ResetCharts", null, null);

firePropertyChange("TextDialog", null, null); //BONUS, repromting user for new input of timer time

if (timerTime > 60 || timerTime == 0){

timesUp = true;

firePropertyChange("Timer", null, "Times Up!");

} else {

timesUp = false;

String newTimer = ("Timer: " + df.format(timerTime) + ".00");

firePropertyChange("Timer", null, newTimer);

}

}

secondsElapsed = 0;

ticks = 0;

recordTicks = 0;

recordCounter = 0;

}

public boolean getTimesUp(){

return timesUp;

}

public int getYRecord(){

return yRecord;

}

public int getYAverage(){

return yAverage;

}

public void setTimerTime(int timerTime){

this.timerTime = timerTime;

}

public void setTimesUp(boolean timesUp){

this.timesUp = timesUp;

}

}